A Model-Based Approach To Requirements Analysis

Eva Geisberger, Johannes Grünbauer, and Bernhard Schätz
Technische Universität München, Institut für Informatik
Boltzmannstr. 3, D-85748 Garching bei München, Germany

Abstract

A major task in designing systems development is the systematic elaboration of functional system requirements and their integration into the environment of the overall technical system. The main challenge is to handle the versatile tasks of coordinating the communication and consolidation of the various stakeholder requirements of the different involved diciplines and derive a common definition of the system behavior, which is appropriate to the problem. The problem- and customer-related product definition must be consolidated with and integrated into the manifold requirements of the functional and technical system design. Accordingly, the model-based requirements analysis and system-definition presented here defines a well-structured modeling approach, which provides a basic model of RE work products (RE Product Model) and systematically guides the goal-oriented formulation and adjustment of the different stakeholder-requirements by using functional system views and descriptive specification techniques. Thus it allows a clear specification of a consistent and complete system design. The central steps of this approach are implemented in a requirements management (RM) tool prototype called AUTORAID.

1. Introduction

The definition of the initial system specification is the source of the most crucial development errors in a development process. To avoid these error, the formulation of a requirements- and systems-specification has to be the result of systematic coordination between the different demands of the stakeholder, the customers and users (users, marketing, distribution, service, product lines) on the one hand, and the technical disciplines like mechanics, electronics and computer science on the other hand. Thus we have to

- analyze the problem and the goals of the product development,
- systematically elaborate the functional requirements and properties, as well as the integration of the system including all its interfaces, and have to describe them precisely. Furthermore, we have to
- elaborate and analyze the manifold constraints which result from the different objectives and the integration into the products and systems, and to
- include the resulting requirements regarding design and realization of the system in an early phase into the specification of the system. As well, we have to consider constraints to the functions, the behavior and the technical realization of the system.

2. Model-Based Requirements Engineering

The central approach of model-based requirements engineering (RE) is the introduction of a common model of specification products – the RE Product Model (Fig. 1). It drives the interdisciplinary elaboration and coordination of the requirements with the aid of elementary models and constraints. It's substantial elements are the definition of *goals* and *strategic constraints*, the resulting *functional requirements* and *general conditions* for the system to be developed from the customer's point of view, and the precise specification of the system concept with its *detailed system requirements* and *constraints* to interfaces and the further design within the disciplines software, mechanics and electronics.

Furthermore, it supports the goal-oriented elaboration and evaluation of requirements and concepts: *Goals* constitute the definition and design of *functional requirements* and *general conditions*, as well as the detailed system concept design. *Functional requirements* describe the user functionalities of the future system, and *general conditions* specify restrictions or design decisions to be observed in the development. These "high-level" requirements have to be refined with aid of the functional modeling of the system and to adjust and complete the concept of the system precisely to the further development. *functional requirements* and *general conditions* are methodically related regarding the design: *general conditions* (in general *non-functional requirements*) are design decisions imposed on the system. They

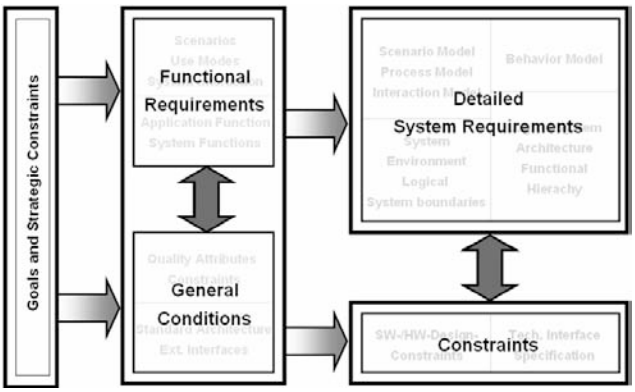

Figure 1. Structure of the RE Product Model

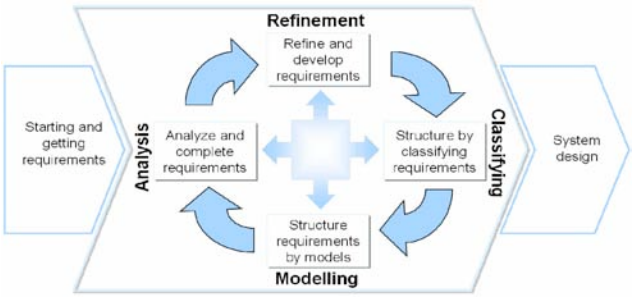

Figure 2. Iterative Process Model in AUTORAID

have to be motivated from business *goals*. They drive the refinement of the "high-level" requirements and the design of the *detailed system concept*, which is systematically derived and specified by functional models.

The modeling concepts provided by the approach describe via five basic "modeling views": **scenario views** (models of the use processes and scenarios), **structural views** (environment model, system boundaries, function hierarchy), **interaction views**, **data views** and **behavioral views**. By the mapping of these views onto a uniform system model, conditions regarding consistence between the views are introduced, which can be used for the review and adjustment of the elaborated requirements- and system-models. Requirements of different stakeholders are mapped on the modeling elements of the system views in a step-wise fashion, are structured, analyzed, and completed with the aid of the underlying system model as well as the consistency conditions. The interactive use of descriptive specifications techniques is substantial for the successful and goal-oriented adjustment of the functional system views.

With the help of this structured modeling approach, a basis for communication and adjustment for the interdisciplinary elaboration, validation, and analysis of a consistent and complete requirements- and system-specification is found. The RE product model allows the common goal-oriented and traceable definition of requirements and serves as a standard for quality and progress control of the specification.

3. Requirements Management – The AUTORAID Tool

The main concept of model-based requirements engineering are reflected in the product model of the AUTORAID[1]-tool (Fig. 5), describing the elements of a system specification and their relation. It contains informal requirements (in form of business and application requirements), classified constraints (architectural, modal, and data type), uses cases (including scenarios as sequences of observations), as well as their relation (in form of association, motivation, and observation) to the concepts of the system model (like component, control state, or EET event).

As explained in the following sections, it guides the multidisciplinary RE activities *Refinement*, *Classifying*, *Modeling*, and *Analysis*, which have to be elaborated in an iterative steps, of requirements analysis and system designs. This iterative feedback loop of refining and completing requirements in AUTORAID is shown in Fig. 2. The step *Starting and getting requirements* shows the steady input of requirements into the process. The activity *system design* shows the adjustment and specification of models/drafts during the whole RE-process.

AUTORAID is integrated into the specification tool AUTOFOCUS [1, 3], and uses its formally founded system views and graphical description techniques (Fig. 3), including *System Structure Diagrams* (SSDs, upper left window) describing the structure of a system, with its components, interfaces, and communication paths; *System Structure Diagrams* (STDs, right window) using extended finite automata to describe the behavior of a component in an SSD; *Data Type Definitions* (DTDs, lower left window) specifying the data types used in the model [9]. *Extended Event Traces* (EETs, right-hand side of Fig. 9) finally make it possible to describe exemplary system runs, similar to MSCs [5]. The verification and simulation support supplied by AUTOFOCUS can be used to validate the requirements.

AUTOFOCUS was developed at the chair of "Systems and Software Engineering" at the TU München as a prototype in a scientific context, and was used successfully in several industrial projects. It connects concepts of system design, simulation, code- and testcase-generation, and provides verification of software components. Fig. 4 shows some of the underlying system concepts – the dependencies between the modeling views of system structure (e. g. *Com-*

[1] AUTOFOCUS Requirements Analysis Integrating Development

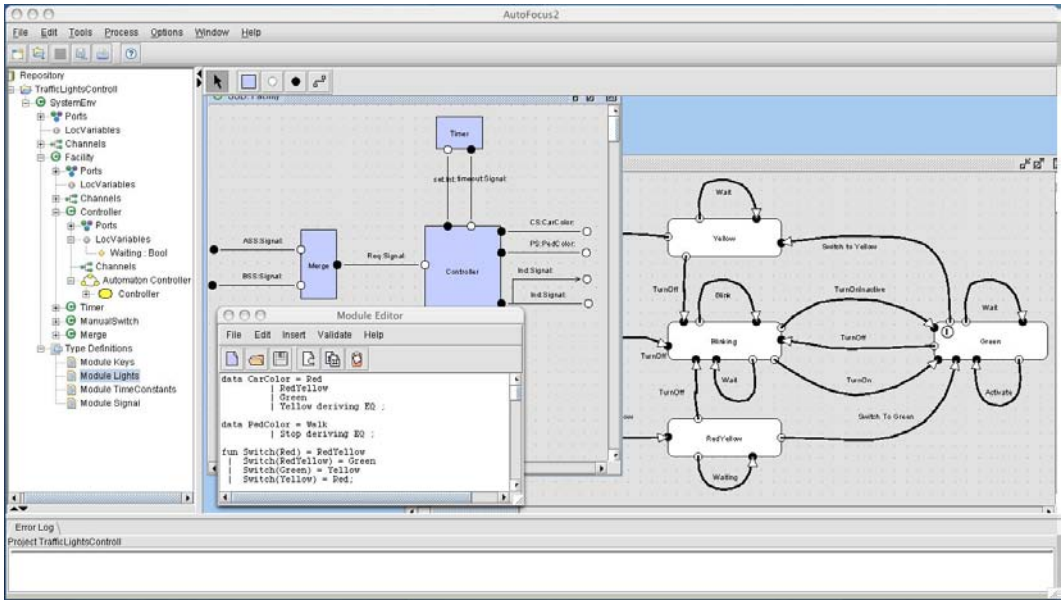

Figure 3. System Views and Representation Techniques in AUTOFOCUS

ponents, Ports and *Channels*) and system behaivior (e. g. *Sequences*, *ControlStates* and *Transitions*).

In the following, we sketch the process of the model-based requirements engineering regarding to the methodological steps of AUTORAID in Fig. 2. A detailed description can be found in [2].

3.1. Getting and Refining Requirements

The requirements engineering process starts with the elicitation of requirements. Common techniques to acquire them are structured workshops or interviews [4, 6]. In the most straightforward case requirements are entered into AUTORAID. To create a new requirement, attributes like *title*, *description*, *status*, *priority*, etc. have to be identified. Fig. 5 shows the concepts within the data model and Fig. 6 shows the concept by the AUTORAID user interface representation. The project tree (lhs. in Fig. 6) shows the refinement structure of requirements. Additionally, requirements source documents and their contexts are integrated into the *Analysis*-tree.

Besides directly creating new requirements, requirements can be created out of a *source document* worked out by a *source context* (meetings, telephone calls, etc.). By "cut-and-paste", requirements can be conveniently created and traced to the source by keeping the link to the corresponding part of the document. AUTORAID supports direct integration of structured requirements documents, e.g, generated by DOORS.

Requirements - distinguished by their unique identifier and title – are added to the *analysis*-branch of a *project*-tree. (cf. Fig. 6, lhs.).

According to the goal-oriented refinement of requirements, *ApplicationRequirements* can be derived from goals (*BusinessRequirements*) in AUTORAID. From *ApplicationRequirements* further *SubApplicationRequirements* can be derived. Vice versa, it is possible to analyze elicited requirements according to their contribution to the goal-achievement, and to structure them in refinement-hierarchies. Fig. 5 shows the corresponding functionality within the AUTORAID data model by the *Is Justified By* relation between *BusinessRequirements* and *Application Requirements*, their *Super-* and *Sub-* relations, and by the different forms of structuring requirements into *UseCases* or *Constraints*. The corresponding menue functions are shown in Fig. 7 (*Edit*, *Classify to*, *Refine*[2], *Create* and *Associate*), and the resulting refinement-structure is represented by the "'goal-trees"' within the *Requirements-* tree. (All *ApplicationRequirements* must be subordinated to *BusinessRequirements*). The refinement relations are also shown in the description of a requirement (right hand side of the AUTORAID window), listing direct *Superrequirements* and *Subrequirements*.

3.2. Classification and Modeling of Requirements

According to the classifying schema of requirements, in AUTORAID requirements can be refined and specified by *UseCases* or *Constraints*. *UseCases* are processes[3],

[2]The sub-menu functions of *Refine* are *Edit Refinement*, *Copy*, *Move*, *Insert* and *Revers*

[3]Business or use processes

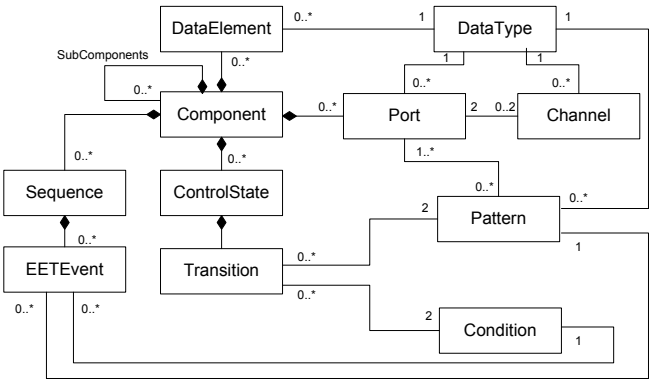

Figure 4. Some Concepts used in AUTOFOCUS

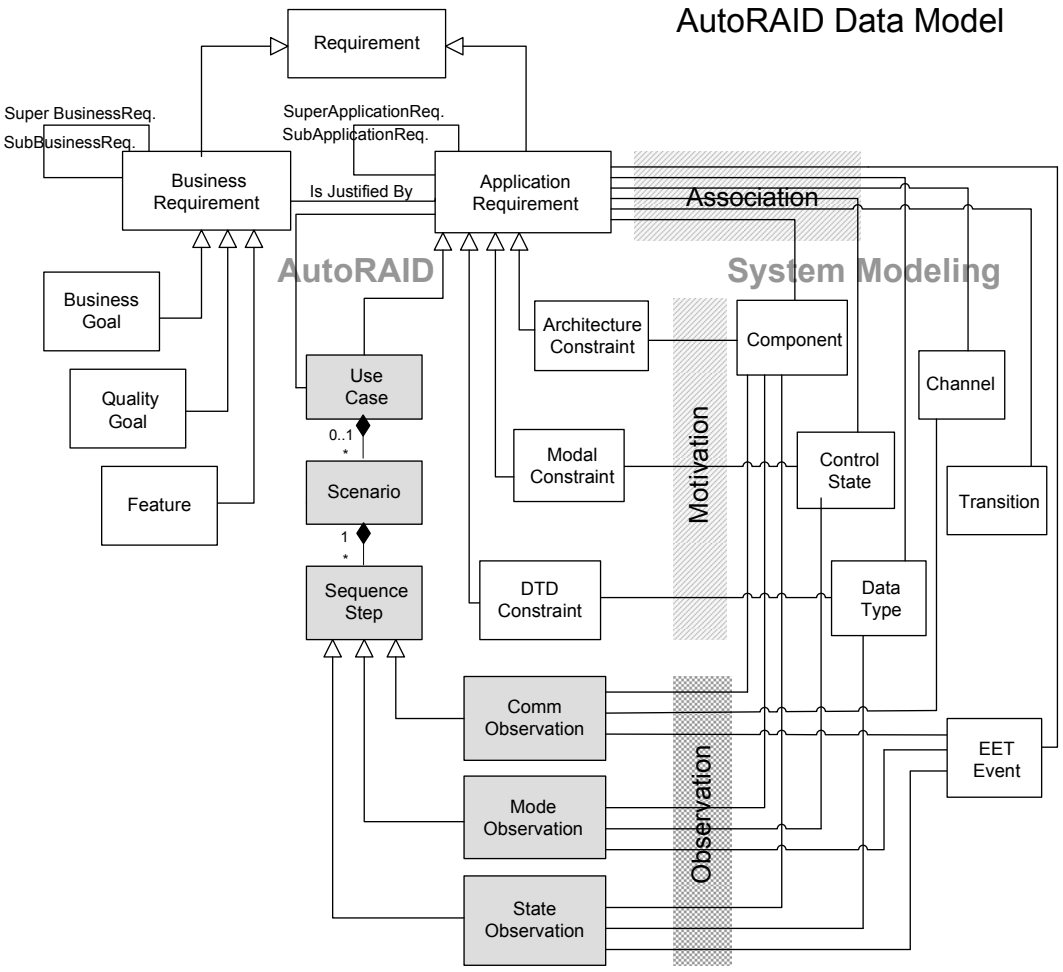

Figure 5. AUTORAID **Data Model**

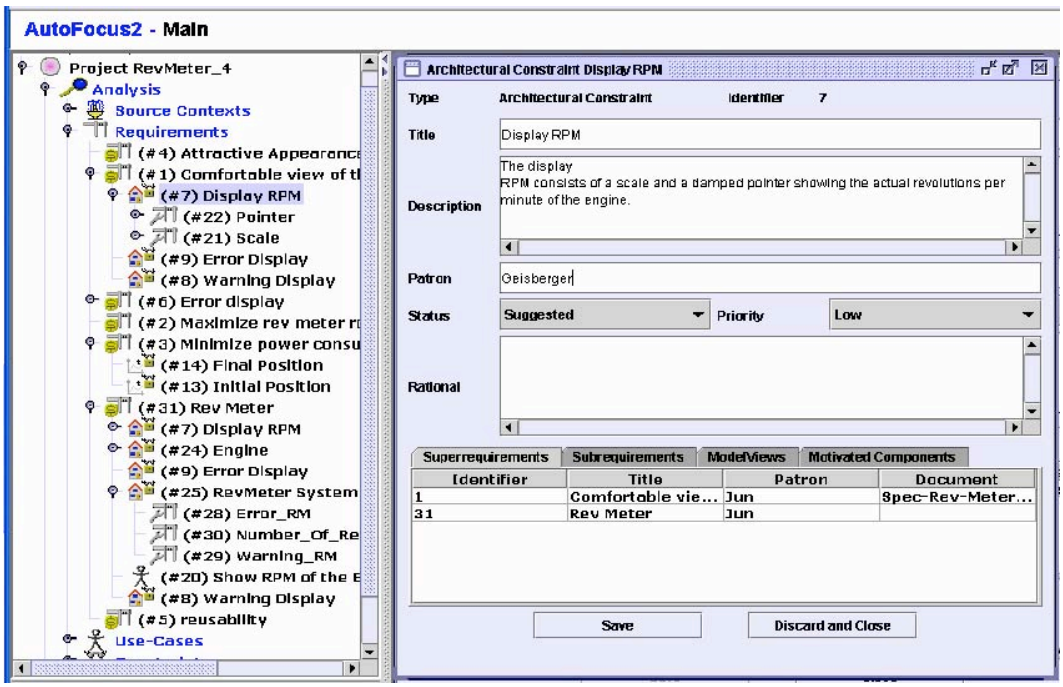

Figure 6. AUTORAID **User Interface**

or required system functions or services, which have to be specified in a more detailed way. *Constraints* are the specification of the system environment or the request for system requirements like architectural-, state- or interface-requirements.

Constraints are separated into (Fig. 5)

- *ArchitecturalConstraints* – requirements regarding the structure of the system to be developed, and its environment. Here, the components, their interfaces and the communication channels can be constituted (structural view).
- *ModalConstraints* – modes of the application. The states and the transitions between them can be defined (state-oriented behavioral view).
- *DTDConstraints* – data type definitions of the communication within the system or a state variable of the modeling of the behavior (data view).

The initial point for the elaboration of functional requirements and system designs is the comprehensive modeling and analysis of both the business- and use-processes of the system. This is done by using of *UseCase*- and *Scenario*-modeling. Starting with the informal use of graphical modeling techniques, like activity diagrams in UML, the main application-process steps and use functions of the system are defined, and modeled within AUTORAID, in an iterative way. This procedure, as well as the detailed analysis and modeling of the scenarios and system interactions are described in Sect. 3.2.3.

Classifying requirements into *Use Case/Scenario*, *Architectural*-, *Modal*- or *DTD-Constraints* is the first step towards detailed system modeling. By and by, the components of the system environment and the system boundaries are defined, the interfaces are sketched and the system functions/services, which have to be developed, are identified. For the purpose of this construction and detailed specification of the component- and system-behavior, in AUTORAID the *Motivation*- and *Association*-function are conceived.

3.2.1 Motivation

The *Motivation*-function in AUTORAID is used to create model elements in AUTOFOCUS out of the *Contraints*-requirements. Fig. 5 shows the corresponding construction relation (*Motivation*) between the requirements for dedicated model elements (*Constraints*) and the system elements to be modeled (*Component*, *Channel*, *State*, *TransitionSegment*, *Type*) of the system-specification-tool AUTOFOCUS.

Fig. 7 shows a corresponding screenshot of AUTORAID, where the motivation and construction of the component *Engine* from the *ArchitecturalConstraint Engine* is demonstrated. It results from the selection of the menu item *Motivate – New Component*. On running this *motivate*-function, in the sub-tree *Modelviews – Architectural* the corresponding component is inserted, and simultaneously in the mod-

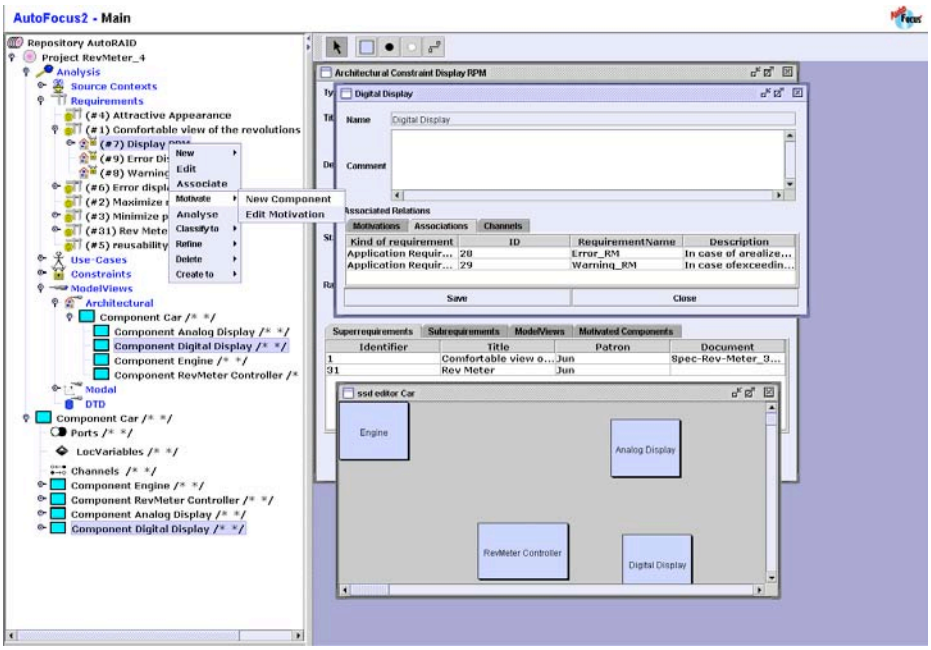

Figure 7. *Motivation-* **and** *Association*-**Functions in** AUTORAID

eling area of AUTOFOCUS the accompanied modeling element is constructed. As a result, the sub-component *Component Engine* is created in the *design*-tree *Component Car* and the component *Engine* is drawn in the graphical SSD modeling view.

This design-relation between requirements and model elements is specified on both sides:

- In the requirements sheet of the *Constraint*-requirement *Architectural Constraint Engine*, the model elements that are *motivated* by this requirement are listed inside the attribute page *Motivated Components* (rear window in the workspace).
- In the attribute sheet of the constructed *Modelview Engine*, the *motivating* items are listed in the *Motivations*-page.

3.2.2 Association

By the *Associations*-relation of the AUTORAID data model previously motivated system model elements (components, channels, modes, scenarios, etc.) can be specified in detail: arbitrary *ApplicationRequirements* can be mapped to the system models and thus specify the system requirements precisely. *Association*. Fig. 7 shows a first mapping of requirements to the system component *RevMeter-Controller* (specification page in the workspace), which has to be developed. The *Associations* are listed in the corresponding page of the attribute-sheet *Digital Display*. For example, for the component *Digital Display* specific error- and warning-displays are required and specified (*Error_RM, Warning_RM*. By this *Association*-function, functional and non-functional requirements can be assigned to *Components*, *UseCases* or *modes*.

Using the *Motivation-* and *Association*-relation of AUTORAID, requirements are worked out, refined and detailed specified by functional system views.

3.2.3 Use-Case and Scenario Analysis

As described before, the basic means to develop and refine functional requirements is a systematic process analysis. Thus, the following tasks have to be done:

- Identification of the main system functions and their application (hierarchical *UseCase* tree with *Scenario* descriptions).
- Elaboration of individual steps performed through these applications (*SequenceSteps*).
- Identification of the relevant components of the overall system (*Motivate ArchitecturalConstraints*).
- Specification of the required modes and system states (*Motivate ModalConstraints*).
- Identification of the necessary communications between, and mode or state changes of the components of the overall system (*CommunicationObservation*, *StateObservation*, and *ModeObservation*).

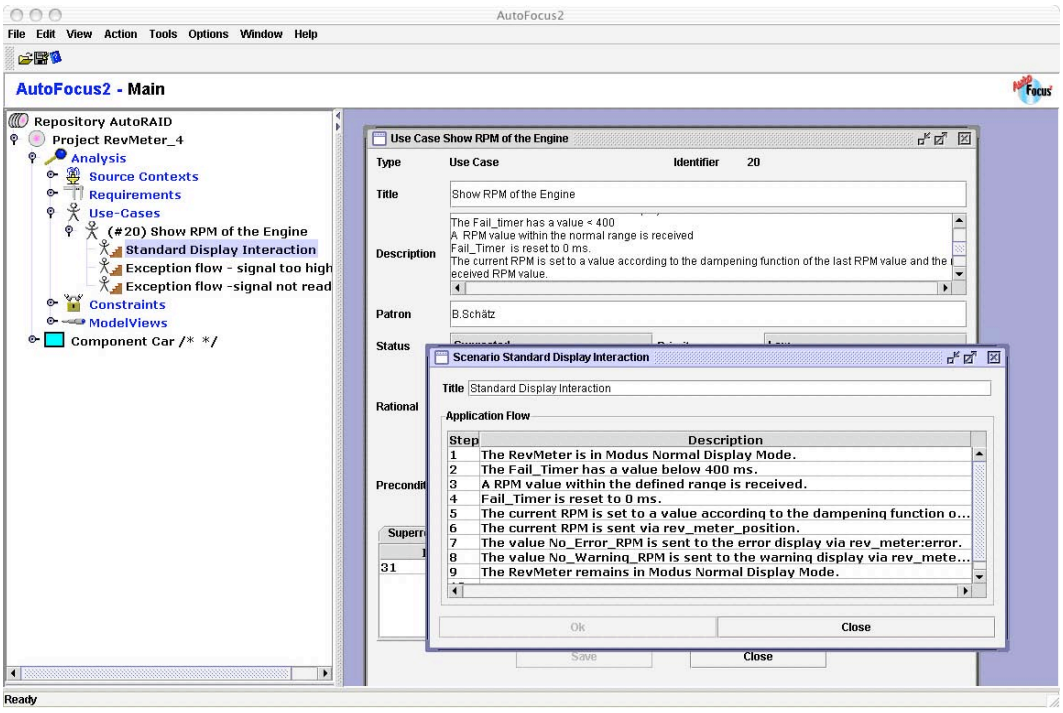

Figure 8. Use Case- and Scenario-Modeling in AUTORAID

Figure 8 shows the corresponding identification of Use Cases and Scenarios in AUTORAID, with several representative scenarios used to detail one Use Case. Furthermore, a single scenario is structured by identification of its steps.

Then, every step of the previously informally described scenarios now can be analyzed and specified according to the system aspects of communication *EventObservation*[4] (left-hand side of Fig. 9), application modes *ModeObservation*, and system/component states *StateObservation*. As illustrated by an *EventObservation* of scenario-step 7, it allows the detailed specification of the *Sender* and *Receiver* components, the communication *Channels* and the *Signal* data structure by selecting from an offered list. If the required component is missing, it has to be constructed by the *Motivation* function before it can be selected within the *EventObservation*. With the options *ModeObservation* or *StateObservation*, the required conditions for the mode and state variables of an interaction step can be specified, respectively.

When the single *Sequencestep* models are defined using the *Observation* analysis, the graphical view of the specified scenario can be generated by the *Generate MSC*-function (see right-hand side of Fig. 9). These scenario models can be used for further analysis or test case generation. According to the *Motivation*-function, by generating the interaction model of a scenario step, a corresponding model element *EET Event* in AUTOFOCUS is created (see Fig. 5). Requirements on these interaction events also can be mapped by the *Association*-function, and therefore structured and specified precisely.

3.3 Completion, Tracing, and Analysis

Major goal of the analysis is to revise requirements and system concepts in terms of product goals and customer needs, and to uncover inconsistencies and ambiguities within the specification. The core techniques of AUTORAID ensuring the developing an adequate specification of the system, are the *constructive support for the refinement* of a specification, the *tracing of requirements*, and a *generic mechanism to analyze* the specification.

Through the constraints of the RE product model, AUTORAID constructively enforces a systematical refinement and completion of the specification. Based on the problem-oriented classification and formalization of requirements, constraints and use cases, objectives must be analyzed systematically (e.g., when identifying sender, receiver, and signal of a communication event). Thus, via the model relations of the product model, 'gaps' and inconsistencies can be found systematically, and then discussed and completed.

By every iteration, the specification gets more and more structured and appropriately modeled, eventually leading – via the model views – to a design specification of the system. Here, based on the goal-oriented refinement-relation,

[4] named *CommObeservation* in the AUTORAID data model

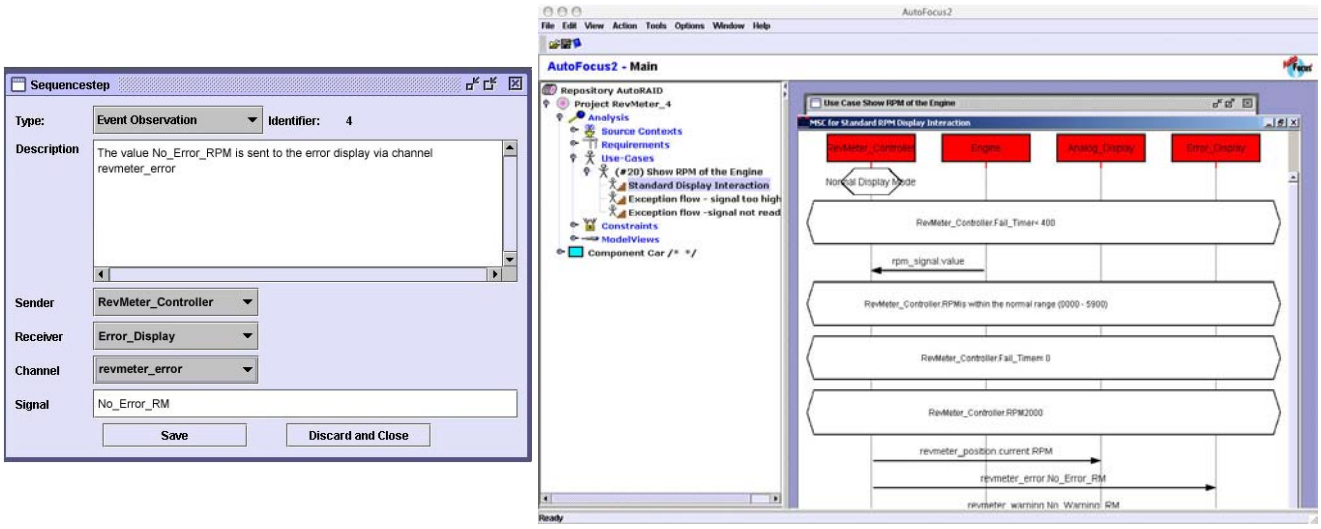

Figure 9. Scenario Analysis in AUTORAID

the benefit of requirements and design decisions can be analyzed, because AUTORAID provides a seamless modeling and coupling of requirements and system design. Based on goals and requirements system models and design concepts can be derived and constructed (*forward tracing*). On the other hand, system design concepts - and respectively solution concepts - can and have to judged regarding its benefit, validated and integrated into the overall system design (*backward tracing*).

While the strictness of model-based formalization step implicitly helps to analyze a specification (e.g., when identifying sender, receiver, and signal of a communication event), analysis techniques in form of consistency conditions can be applied to detected possible weaknesses of the model. Some of these consistency, or rather, soundness conditions used in the AUTORAID approach are:

- Each business requirement must be refined by at least on application requirement.
- Each application requirement must be classified or refined by a further application requirement.
- Each classified requirements must be formalized by a element of the design level.

While the structuring, classification, and formalization steps are performed with user interaction, assisted by convenient support for fast and efficient creation of sub-specifications, model-elements, etc., the consistency analysis is performed automatically, presenting those specification and model elements that do not meet the consistency conditions.

4. Related Work

The basis of the AUTORAID approach is the elaboration, structuring, analysis, and validation and completion of requirements with the aid of basic functional models, as well as the consequent deduction of the requirements and system specification from goals (forward- and backward tracing). With the realization of this concepts within the tool AUTORAID and its integration into the mathematically sound specification tool AUTOFOCUS, its possibilities of verification can be used for validation and completion of the requirements.

Approaches of requirements structuring with the help of functional models can be found in the VORD approach [7], the KAOS approach [10] and in Leite's and Freeman's work on *Requirements Validation Through Viewpoint Resolution* (cf. [8]).

The root of VORD is the analysis and structuring of requirements in the view of features (services). The services of a system are described using scenarios. Non-functional requirements are mapped to these services. The services can be specified with *event traces* and state automata. The tool support of VORD allows to systematically recognizing conflicts between requirements of different operational viewpoints (service specifications). A validation of the requirements with the mapping to mathematically founded models and a tool supported verification is not provided by VORD so far.

Leite and Freeman [8] structure requirements using different view points in basic models: data view, process view and architectural view. They provide heuristics to discover conflicts within different requirements. A mapping of this generic concept of structuring requirements to precise and

mathematically founded information with the possibility of verification is not given here either.

Goal-oriented approaches like KAOS [10] analyze and elaborate requirements (goals) with aid of top-down and bottom-up solutions. Additionally, KAOS defines a methodological concept for refinement of goals and mapping of the gained detailed requirements to software components (agents). Here, a concept for structuring requirements into functional (goals), non-functional requirements (softgoals) and a further consideration of the requirements regarding their relations (AND/OR-structures) are proposed. If the requirements regarding agents are derived, they can be specified precisely and verified with help of temporal logic. A stepwise elaboration and structuring of requirements with the aid of functional models is not provided by KAOS. Here, a gap exists between the functional structuring and the detailed requirements regarding agents.

In [4] a review-based approach for the stepwise structuring of textual requirements is proposed. This works with use-case descriptions structured using tables, and with state chart diagrams. AUTORAID simplifies this review process with the tool-based support of single transformation steps and provides methods for analysis, as well as the consideration of other aspects (e. g. data and structure).

In contrast to state-of-the-art RE tools like DOORS, Requiste Pro, or Caliber, which provide a generic product model consisting only of hierarchic and linked requirements, AUTORAID uses a rich, domain-specific product model with specific concepts like scenarios, modes, or observations. Therefore, it effectively supports a multi-stakeholder, review-based RE process, improving the quality of a requirements-specification in the early development steps as well as easing the transition to the design model.

5. Summary and Outlook

AUTORAID provides a model-based requirements analysis by a reference model of RE work products (RE Product Model) and a structured and stepwise transition from textual requirements to a design model. The goal-oriented Product Model provides a communication base for interdisciplinary consolidation of requirements and guides the iterative refinement and completion to a problem-adequate system specifications. It contains a detailed conceptual model with different classes of requirements (e. g. business and application requirements, use cases, architectural constraints,modal constraints) and tool-supported steps for integrating requirements analysis and functional system design. The final specified system behavior best meets the business, user and quality goals of the development. Actual work extends the approach to a general model-based RE reference model that is tailorable to specific domain or project needs. Using a product model with extended, domain specific requirements and views (e. g. time constraints), a deep structuring and strong interconnection between requirements and system model views gets possible. Major goal is the assistance of complex analysis techniques (e. g. checking the consistency of system use scenarios and its state-based behavior specification or verifying the consistency of architectural interfaces constraints and functional system requirements), and the support of detailed generative steps like generating test cases from structured application scenarios.

References

[1] AUTOFOCUS Homepage, Documentation, Screenshots, Tutorials, and Downloads. `http://autofocus.in.tum.de`.

[2] AUTORAID Homepage, Documentation, Screenshots, and Downloads. `http://wwwbroy.in.tum.de/~autoraid/`.

[3] P. Braun, H. Lötzbeyer, B. Schätz, and O. Slotosch. Consistent Integration of Formal Methods. In *Proc. 6th Intl. Conf on Tools for the Analysis of Correct Systems (TACAS), LNCS 2280*, 2000.

[4] C. Denger and B. Paech. An Integrated Quality Assurance Approach for Use Case Based Requirements. In B. Rumpe and W. Hesse, editors, *Proceedings zur Tagung Modellierung 2004*, pages 307–308, Marburg, Germany, March, 24–26 2004.

[5] ITU. ITU-TS Recommendation Z.120: Message Sequence Chat (MSC). ITU-TS, Geneva 1996.

[6] G. Kontonya and I. Sommerville. *Requirements Engineering: Processes and Techniques*. Wiley & Sons, 1998.

[7] G. Kotonya and I. Sommerville. Requirements Engineering with Viewpoints. Technical report, Lancester University, 1995.

[8] L. Leite and P. Freeman. Requirements Validation Through Viewpoint Resolution. *IEEE Transaction on Software Engineering*, 1991.

[9] J. Phillips and O. Slotosch. The Quest for Correct Systems: Model Checking of Diagrams and Datatypes. In *Asia Pacific Software Engineering Conference*, 1999.

[10] A. van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. In *Proceedings of the 5th IEEE International Symposium on Requirements Engineering*, pages 249–263, Toronto, August 2001.